# DIRT – Discovery of Inference Rules from Text

Dekang Lin and Patrick Pantel
University of Alberta
Department of Computing Science
Edmonton, Alberta T6H 2E1 Canada
{lindek, ppantel}@cs.ualberta.ca

## ABSTRACT

In this paper, we propose an unsupervised method for discovering inference rules from text, such as "*X is author of Y ≈ X wrote Y*", "*X solved Y ≈ X found a solution to Y*", and "*X caused Y ≈ Y is triggered by X*". Inference rules are extremely important in many fields such as natural language processing, information retrieval, and artificial intelligence in general. Our algorithm is based on an extended version of Harris' Distributional Hypothesis, which states that words that occurred in the same contexts tend to be similar. Instead of using this hypothesis on words, we apply it to paths in the dependency trees of a parsed corpus.

## 1. INTRODUCTION

Text is the most significant repository of human knowledge. Many algorithms have been proposed to mine textual data. Most of them focus on document clustering [13], identifying prototypical documents [20], or finding term associations [14] and hyponym relationships [9]. We propose an unsupervised method for discovering inference rules, such as "*X is author of Y ≈ X wrote Y*", "*X solved Y ≈ X found a solution to Y*", and "*X caused Y ≈ Y is triggered by X*". Inference rules are extremely important in many fields such as natural language processing, information retrieval, and artificial intelligence in general.

For example, consider the query to an information retrieval system: "*Who is the author of the 'Star Spangled Banner'?*" Unless the system recognizes the relationship between "*X wrote Y*" and "*X is the author of Y*", it would not necessarily rank the sentence

> ... Francis Scott Key wrote the "Star Spangled Banner" in 1814.

higher than the sentence

> …comedian-actress Roseanne Barr sang her famous shrieking rendition of the "Star Spangled Banner" before a San Diego Padres-Cincinnati Reds game.

We call "*X wrote Y ≈ X is the author of Y*" an inference rule. In previous work, such relationships have been referred to as paraphrases or variants [24]. In this paper, we use the term **inference rule** because we also want to include relationships that are not exactly paraphrases, but are nonetheless related and are potentially useful to information retrieval systems. For example, "*X caused Y ≈ Y is blamed on X*" is an inference rule even though the two phrases do not mean exactly the same thing.

Traditionally, knowledge bases containing such inference rules are created manually. This knowledge engineering task is extremely laborious. More importantly, building such a knowledge base is inherently difficult since humans are not good at generating a complete list of rules. For example, while it is quite trivial to come up with the rule "*X wrote Y ≈ X is the author of Y*", it seems hard to dream up a rule like "*X manufactures Y ≈ X's Y factory*", which can be used to infer that "*Chrétien visited Peugot's newly renovated car factory in the afternoon*" contains an answer to the query "*What does Peugot manufacture?*"

Most previous efforts on knowledge engineering have focused on creating tools for helping knowledge engineers transfer their knowledge to machines [6]. Our goal is to automatically discover such rules.

In this paper, we present an unsupervised algorithm, **DIRT**, for **Discovery of Inference Rules from Text**. Our algorithm is a generalization of previous algorithms for finding similar words [10][15][19]. Algorithms for finding similar words assume the Distributional Hypothesis, which states that words that occurred in the same contexts tend to have similar meanings [7]. Instead of applying the Distributional Hypothesis to words, we apply it to paths in dependency trees. Essentially, if two paths tend to link the same sets of words, we hypothesize that their meanings are similar. Since a path represents a binary relationship, we generate an inference rule for each pair of similar paths.

The remainder of this paper is organized as follows. In the next section, we review previous work. In Section 3, we define paths in dependency trees and describe their extraction from a parsed corpus. Section 4 presents the DIRT system and a comparison of our system's output with manually generated paraphrase expressions is shown in Section 5. Finally, we conclude with a discussion of future work.

## 2. Previous Work

Most previous work on variant recognition and paraphrase has been done in the fields of natural language generation, text summarization, and information retrieval.

The generation community has focused mainly on rule-based text transformations in order to meet external constraints such as length and readability [11][18][22]. Dras [4] described syntactic


KDD 01 San Francisco CA USA

paraphrases using a meta-grammar with a synchronous Tree Adjoining Grammar (TAG) formalism.

In multi-document summarization, paraphrasing is important to avoid redundant statements in a summary. Given a collection of similar sentences (a theme) with different wordings, it is difficult to identify similar phrases that report the same fact. Barzilay et al. [3] analyzed 200 two-sentence themes from a corpus and extracted seven lexico-syntactic paraphrasing rules. These rules covered 82% of syntactic and lexical paraphrases, which cover 70% of all variants. The rules are subsequently used to identify common statements in a theme by comparing the predicate-argument structure of the sentences within the theme.

In information retrieval, it is common to identify phrasal terms from queries and generate their variants for query expansion. It has been shown that such query expansion does promote effective retrieval [1][2]. Morphological variant query expansion was treated by Sparck Jones and Tait [24] and Jacquemin [12].

In [21], Richardson extracted semantic relationships (e.g., hypernym, location, material and purpose) from dictionary definitions using a parser and constructed a semantic network. He then described an algorithm that uses paths in the semantic network to compute the similarity between words. In a sense, our algorithm is a dual of Richardson's approach. While Richardson used paths as features to compute the similarity between words, we use words as features to compute the similarity of paths.

Many text mining algorithms aim at finding association rules between terms [14]. In contrast, the output of our algorithm is a set of associations between relations. Term associations usually require human interpretation. Some of them are considered to be uninterpretable even by humans [5].

# 3. Extraction of Paths

The inference rules discovered by DIRT are between paths in dependency trees. In this section, we introduce dependency trees and define paths in trees. Finally, we describe an algorithm for extracting paths from the trees.

## 3.1 Dependency Trees

A dependency relationship [8] is an asymmetric binary relationship between a word called **head**, and another word called **modifier**. The structure of a sentence can be represented by a set of dependency relationships that form a tree. A word in the sentence may have several modifiers, but each word may modify at most one word. The root of the dependency tree does not modify any word. It is also called the head of the sentence.

For example, Figure 1 shows the dependency tree for the sentence "*John found a solution to the problem*", generated by a broad-coverage English parser called Minipar[1] [16]. The links in the diagram represent dependency relationships. The direction of a link is from the head to the modifier in the relationship. Labels associated with the links represent types of dependency relations. Table 1 lists a subset of the dependency relations in Minipar outputs.

[1]Available at www.cs.ualberta.ca/~lindek/minipar.htm.

**Table 1. A subset of dependency relations in Minipar outputs.**

| RELATION | DESCRIPTION | EXAMPLE |
|---|---|---|
| appo | appositive of a noun | the CEO, **John** |
| det | determiner of a noun | **the** dog |
| gen | genitive modifier of a noun | **John's** dog |
| mod | adjunct modifier of any head | **tiny** hole |
| nn | prenominal modifier of a noun | **station** manager |
| subj | subject of a verb | **John** loves Mary. |

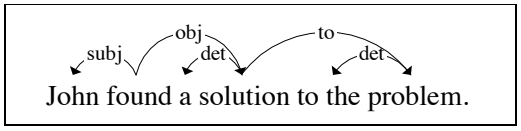



**Figure 1. Example dependency tree.**

Minipar parses newspaper text at about 500 words per second on a Pentium-III 700Mhz with 500MB memory. Evaluation with the manually parsed SUSANNE corpus [23] shows that about 89% of the dependency relationships in Minipar outputs are correct.

## 3.2 Paths in Dependency Trees

In the dependency trees generated by Minipar, each link between two words in a dependency tree represents a direct semantic relationship. A path allows us to represent indirect semantic relationships between two content words. We name a path by concatenating dependency relationships and words along the path, excluding the words at the two ends. For the sentence in Figure 1, the path between *John* and *problem* is named: N:subj:V←find→V:obj:N→solution→N:to:N (meaning "*X finds solution to Y*"). The reverse path can be written as: N:to:N←solution←N:obj:V←find→ V:subj:N. The **root** of both paths is *find*. A path begins and ends with two dependency relations. We call them the two slots of the path: *SlotX* on the left-hand side and *SlotY* on the right-hand side. The words connected by the path are the fillers of the slots. For example, *John* fills the *SlotX* of N:subj:V←find→V:obj:N→solution→N:to:N and *problem* fills the *SlotY*. The reverse is true for N:to:N←solution←N:obj:V←find→V:subj:N. In a path, dependency relations that are not slots are called **internal relations**. For example, find→V:obj:N→solution is an internal relation in the previous path.

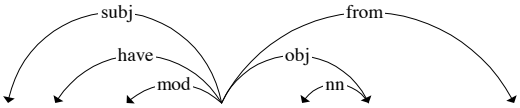
We impose a set of constraints on the paths to be extracted from text for the following reasons:

- most meaningful inference rules involve only paths that satisfy these conditions;
- the constraints significantly reduce the number of distinct paths and, consequently, the amount of computation required for computing similar paths (described in Section 4.3); and
- the constraints alleviate the sparse data problem because long paths tend to have very few occurrences.

The constraints we impose are:

- slot fillers must be nouns because slots correspond to variables in inference rules and we expect the variables to be instantiated by entities;
- any dependency relation that does not connect two content words (i.e. nouns, verbs, adjectives or adverbs) is excluded from a path. E.g. in Figure 1, the relation between *a* and *solution* is excluded;
- the frequency count of an internal relation must exceed a threshold; and

Consider the following sentence:



They had previously bought bighorn sheep from Comstock.

The paths extracted from this sentence and their meanings are:

(a) N:subj:V←buy→V:from:N
≡ *X* buys something from *Y*

(b) N:subj:V←buy→V:obj:N
≡ *X* buys *Y*

(c) N:subj:V←buy→V:obj:N→sheep→N:nn:N
≡ *X* buys *Y* sheep

(d) N:nn:N←sheep←N:obj:V←buy→V:from:N
≡ *X* sheep is bought from *Y*

(e) N:obj:V←buy→V:from:N
≡ *X* is bought from *Y*

An inverse path is also added for each one above.

## 4. Discovering Inference Rules from Text

A path is a binary relation between two entities. In this section, we present an algorithm, called DIRT, to automatically discover the inference relations between such binary relations.

### 4.1 Underlying Assumption

Most algorithms for computing word similarity from text corpus are based on a principle known as the Distributional Hypothesis [7]. The idea is that words that tend to occur in the same contexts tend to have similar meanings. Previous efforts differ in their representation of the context and in their formula for computing the similarity between two sets of contexts. Some algorithms use the words that occurred in a fixed window of a given word as its context while others use the dependency relationships of a given word as its context [15]. Consider the words *duty* and *responsibility*. There are many contexts in which both of these words can fit. For example,

- *duty* can be modified by adjectives such as *additional*, *administrative*, *assigned*, *assumed*, *collective*, *congressional*, *constitutional*, …, so can *responsibility*;
- *duty* can be the object of verbs such as *accept*, *articulate*, *assert*, *assign*, *assume*, *attend to*, *avoid*, *become*, *breach*, …, so can *responsibility*.

**Table 2. Sample slot fillers for two paths extracted from a newspaper corpus.**

| "*X finds a solution to Y*" | | "*X solves Y*" | |
|---|---|---|---|
| *SLOTX* | *SLOTY* | *SLOTX* | *SLOTY* |
| commission | strike | committee | problem |
| committee | civil war | clout | crisis |
| committee | crisis | government | problem |
| government | crisis | he | mystery |
| government | problem | she | problem |
| he | problem | petition | woe |
| legislator | budget deficit | researcher | mystery |
| sheriff | dispute | sheriff | murder |

Based on these common contexts, one can statistically determine that *duty* and *responsibility* have similar meanings.

In the algorithms for finding word similarity, dependency links are treated as contexts of words. In contrast, our algorithm for finding inference rules treats the words that fill the slots of a path as a context for the path. We make an assumption that this is an extension to the Distributional Hypothesis:

> **Extended Distributional Hypothesis:**
> If two paths tend to occur in similar contexts, the meanings of the paths tend to be similar.

For example, Table 2 lists a set of example pairs of words connected by the paths N:subj:V←find→V:obj:N→solution→N:to:N ("*X finds a solution to Y*") and N:subj:V←solve→V:obj:N ("*X solves Y*"). As it can be seen from the table, there are many overlaps between the corresponding slot fillers of the two paths. By the Extended Distributional Hypothesis, we can then claim that the two paths have similar meaning.

### 4.2 Triples

To compute the path similarity using the Extended Distributional Hypothesis, we need to collect the frequency counts of all paths in a corpus and the slot fillers for the paths. For each instance of a path $p$ that connects two words $w_1$ and $w_2$, we increase the frequency counts of the two triples ($p$, *SlotX*, $w_1$) and ($p$, *SlotY*, $w_2$). We call (*SlotX*, $w_1$) and (*SlotY*, $w_2$) features of path $p$. Intuitively, the more features two paths share, the more similar they are.

We use a **triple database** (a hash table) to accumulate the frequency counts of all features of all paths extracted from a parsed corpus. An example entry in the triple database for the path

N:subj:V←pull→V:obj:N→body→N:from:N
≡ "*X pulls body from Y*"

is shown in Figure 2. The first column of numbers in Figure 2 represents the frequency counts of a word filling a slot of the path and the second column of numbers is the mutual information

| X pulls body from Y: | | | |
|---|---|---|---|
| *SlotX*: | | | |
| | diver | 1 | 2.45 |
| | equipment | 1 | 1.65 |
| | police | 2 | 2.24 |
| | rescuer | 3 | 4.84 |
| | resident | 1 | 1.60 |
| | who | 2 | 1.32 |
| | worker | 1 | 1.37 |
| *SlotY*: | | | |
| | bus | 2 | 3.09 |
| | coach | 1 | 2.05 |
| | debris | 1 | 2.36 |
| | feet | 1 | 1.75 |
| | hut | 1 | 2.73 |
| | landslide | 1 | 2.39 |
| | metal | 1 | 2.09 |
| | wreckage | 3 | 4.81 |

**Figure 2. An example entry in the triple database for the path "X pulls body from Y".**

between a slot and a slot filler. Mutual information measures the strength of the association between a slot and a filler. We explain mutual information in detail in Section 4.3. The triple database records the fillers of *SlotX* and *SlotY* separately. Looking at the database, one would be unable to tell which *SlotX* filler occurred with which *SlotY* filler in the corpus.

## 4.3 Similarity between Two Paths

Once the triple database is created, the similarity between two paths can be computed in the same way that the similarity between two words is computed in [15]. Essentially, two paths have high similarity if there are a large number of common features. However, not every feature is equally important. For example, the word *he* is much more frequent than the word *sheriff*. Two paths sharing the feature (*SlotX*, *he*) is less indicative of their similarity than if they shared the feature (*SlotX*, *sheriff*). The similarity measure proposed in [15] takes this into account by computing the mutual information between a feature and a path.

We use the notation $|p, SlotX, w|$ to denote the frequency count of the triple ($p$, *SlotX*, $w$), $|p, SlotX, *|$ to denote $\sum_w |p, SlotX, w|$, and $|*, *, *|$ to denote $\sum_{p,s,w} |p, s, w|$.

Following [15], the mutual information between a path slot and its filler can be computed by the formula:

$$mi(p, Slot, w) = \log\left(\frac{|p, Slot, w| \times |*, Slot, *|}{|p, Slot, *| \times |*, Slot, w|}\right) \quad \textbf{(1)}$$

The similarity between a pair of slots: $slot_1 = (p_1, s)$ and $slot_2 = (p_2, s)$, is defined as:

$$sim(slot_1, slot_2) = \frac{\sum_{w \in T(p_1,s) \cap T(p_2,s)} mi(p_1, s, w) + mi(p_2, s, w)}{\sum_{w \in T(p_1,s)} mi(p_1, s, w) + \sum_{w \in T(p_2,s)} mi(p_2, s, w)} \quad \textbf{(2)}$$

**Table 3. The top-20 most similar paths to "X solves Y".**

| | |
|---|---|
| *Y* is solved by *X* | *Y* is resolved in *X* |
| *X* resolves *Y* | *Y* is solved through *X* |
| *X* finds a solution to *Y* | *X* rectifies *Y* |
| *X* tries to solve *Y* | *X* copes with *Y* |
| *X* deals with *Y* | *X* overcomes *Y* |
| *Y* is resolved by *X* | *X* eases *Y* |
| *X* addresses *Y* | *X* tackles *Y* |
| *X* seeks a solution to *Y* | *X* alleviates *Y* |
| *X* do something about *Y* | *X* corrects *Y* |
| *X* solution to *Y* | *X* is a solution to *Y* |

where $p_1$ and $p_2$ are paths, $s$ is a slot, $T(p_i, s)$ is the set of words that fill in the $s$ slot of path $p_i$.

The similarity between a pair of paths $p_1$ and $p_2$ is defined as the geometric average of the similarities of their *SlotX* and *SlotY* slots:

$$S(p_1, p_2) = \sqrt{sim(SlotX_1, SlotX_2) \times sim(SlotY_1, SlotY_2)} \quad \textbf{(3)}$$

where $SlotX_i$ and $SlotY_i$ are path $i$'s *SlotX* and *SlotY* slots.

## 4.4 Finding the Most Similar Paths

The discovery of inference rules is made by finding the most similar paths of a given path. The challenge here is that there are a large number of paths in the triple database. The database used in our experiments contains over 200,000 distinct paths. Computing the similarity between every pair of paths is obviously impractical.

Given a path $p$, our algorithm for finding the most similar paths of $p$ takes three steps:

(a) Retrieve all the paths that share at least one feature with $p$ and call them candidate paths. This can be done efficiently by storing for each word the set of slots it fills in.

(b) For each candidate path $c$, count the number of features shared by $c$ and $p$. Filter out $c$ if the number of its common features with $p$ is less than a fixed percent (we used 1%) of the total number of features for $p$ and $c$. This step effectively uses a simpler similarity formula to filter out some of the paths since computing mutual information is more costly than counting the number of features. This idea has previously been used in Canopy [17].

(c) Compute the similarity between $p$ and the candidates that passed the filter using equation (2) and output the paths in descending order of their similarity to $p$.

Table 3 lists the Top-50 most similar paths to "*X solves Y*" generated by DIRT. Most of the paths can be considered as paraphrases of the original expression.

# 5. Experimental Results

We performed an evaluation of our algorithm by comparing the inference rules it generates with a set of human-generated paraphrases of the first six questions in the TREC-8 Question-

**Table 4. First six questions from TREC-8.**

| Q# | QUESTION |
|---|---|
| $Q_1$ | Who is the author of the book, "The Iron Lady: A Biography of Margaret Thatcher"? |
| $Q_2$ | What was the monetary value of the Nobel Peace Prize in 1989? |
| $Q_3$ | What does the Peugeot company manufacture? |
| $Q_4$ | How much did Mercury spend on advertising in 1993? |
| $Q_5$ | What is the name of the managing director of Apricot Computer? |
| $Q_6$ | Why did David Koresh ask the FBI for a word processor? |

Answering Track, listed in Table 4. TREC (Text REtrievial Conference) is a U.S. government sponsored competition on information retrieval held annually since 1992. In the Question-Answering Track, the task for participating systems is to find answers to natural-language questions like those in Table 4.

## 5.1 Results

We used Minipar to parse about 1GB of newspaper text (AP Newswire, San Jose Mercury, and Wall Street Journal). Using the methods discussed in Section 3, we extracted 7 million paths from the parse trees (231,000 unique) and stored them in a triple database.

The second column of Table 5 shows the paths that we identified from the TREC-8 questions. For some questions, more than one path was identified. For others, no path was found. We compare the output of our algorithm with a set of manually generated paraphrases of the TREC-8 questions made available at ISI[2].

We also extracted paths from the manually generated paraphrases. For some paraphrases, an identical path is extracted. For example, "*What things are manufactured by Peugeot?*" and "*What products are manufactured by Peugeot?*" both map to the path "*X is manufactured by Y*". The number of paths for the manually generated paraphrases of TREC-8 questions is shown in the third column of Table 5.

For each of the paths $p$ in the second column of Table 5, we ran the DIRT algorithm to compute its Top-40 most similar paths using the triple database. We then manually inspected the outputs and classified each extracted path as *correct* or *incorrect*. A path $p'$ is judged correct if a sentence containing $p'$ might contain an answer to the question from which $p$ was extracted. Consider question $Q_3$ in Table 4 where we have $p$ = "*X manufactures Y*" and we find $p'$ = "*X's Y factory*" as one of $p$'s Top-40 most similar paths. Since "*Peugeot's car factory*" might be found in some corpus, $p'$ is judged correct. Note that not all sentences containing $p'$ necessarily contain an answer to $Q_3$ (e.g. "*Peugeot's Sochaux factory*" gives the location of a Peugeot factory in France). The fourth column in Table 5 shows the number of Top-40 most similar paths classified as *correct* and the fifth column gives the intersection between columns three and four. Finally, the last column in Table 5 gives the percentage of correctly classified paths.

[2] Available at http://www.isi.edu/~gerber/Variations2.txt

**Table 5. Evaluation of Top-40 most similar paths.**

| Q# | PATHS | MAN. | DIRT | INT. | ACC. |
|---|---|---|---|---|---|
| $Q_1$ | X is author of Y | 7 | 21 | 2 | 52.5% |
| $Q_2$ | X is monetary value of Y | 6 | 0 | 0 | N/A |
| $Q_3$ | X manufactures Y | 13 | 37 | 4 | 92.5% |
| $Q_4$ | X spend Y | 7 | 16 | 2 | 40.0% |
| | spend X on Y | 8 | 15 | 3 | 37.5% |
| $Q_5$ | X is managing director of Y | 5 | 14 | 1 | 35.0% |
| $Q_6$ | X asks Y | 2 | 23 | 0 | 57.5% |
| | asks X for Y | 2 | 14 | 0 | 35.0% |
| | X asks for Y | 3 | 21 | 3 | 52.5% |

## 5.2 Observations

There is very little overlap between the automatically generated paths and the paraphrases, even though the percentage of correct paths in DIRT outputs can be quite high. This suggests that finding potentially useful inference rules is very difficult for humans as well as machines. Table 6 shows some of the *correct* paths among the Top-40 extracted by our system for two of the TREC-8 questions. Many of the variations generated by DIRT that are correct paraphrases are missing from the manually generated variations. It is difficult for humans to recall a list of paraphrases. However, given the output of our system, humans can easily identify the correct inference rules. Hence, at the least, our system would greatly ease the manual construction of inference rules for an information retrieval system.

The performance of DIRT varies a great deal for different paths. Usually, the performance for paths with verb roots is much better than for paths with noun roots. A verb phrase typically has more than one modifier, whereas nouns usually take a smaller number of modifiers. When a word takes less than two modifiers, it will not be the root of any path. As a result, paths with noun roots occur less often than paths with verb roots, which explains the lower performance with respect to paths with noun roots.

In Table 5, DIRT found no correct inference rules for $Q_2$. This is due to the fact that $Q_2$ does not have any entries in the triple database.

## 6. Conclusion and Future Work

Better tools are necessary to tap into the vast amount of textual data that is growing at an astronomical pace. Knowledge about inference relationships in natural language expressions would be extremely useful for such tools. To the best of our knowledge, this is the first attempt to discover such knowledge automatically from a large corpus of text. We introduced the Extended Distributional Hypothesis, which states that paths in dependency trees have similar meanings if they tend to connect similar sets of words. Treating paths as binary relations, our algorithm is able to generate inference rules by searching for similar paths. Our experimental results show that the Extended Distributional Hypothesis can indeed be used to discover very useful inference

**Table 6. Paths found for two of the six questions in TREC-8 and the variations discovered manually and by DIRT.**

| | $Q_1$ | $Q_3$ |
|---|---|---|
| PATHS | *X* is author of *Y* | *X* manufactures *Y* |
| MANUAL VARIATIONS | *Y* is the work of *X*; *X* is the writer of *Y*; *X* penned *Y*; *X* produced *Y*; *X* authored *Y*; *X* chronicled *Y*; *X* wrote *Y* | *X* makes *Y*; *X* produce *Y*; *X* is in *Y* business; *Y* is manufactured by *X*; *Y* is provided by *X*; *Y* is *X*'s product; *Y* is product from *X*; *Y* is *X* product; *Y* is product made by *X*; *Y* is example of *X* product; *X* is manufacturer of *Y*; find *Y* in *X*'s product line; find *Y* in *X* catalog |
| DIRT VARIATIONS | X co-authors Y; X is co-author of Y; X writes Y; X edits Y; Y is co-authored by X; Y is authored by X; X tells story in Y; X translates Y; X writes in Y; X notes in Y; … | X produces Y; X markets Y; X develops Y; X is supplier of Y; X ships Y; X supplies Y; Y is manufactured by X; X is maker of Y; X introduces Y; X exports Y; X makes Y; X builds Y; X's production of Y; X unveils Y; Y is bought from X; X's line of Y; X assembles Y; X is Y maker; X's Y factory; X's Y production; X is manufacturer of Y; X's Y division; X meets demand for Y; … |

rules, many of which, though easily recognizable, are difficult for humans to recall.

Many questions remain to be addressed. One is to recognize the polarity in inference relationships. High similarity values are often assigned to relations with opposite polarity. For example, "*X worsens Y*" has one of the highest similarity to "*X solves Y*" according to equation (2). In some situations, this may be helpful while for others it may cause confusion.

Another is to extend paths with constraints on the inference rule's variables. For example, instead of generating a rule "*X manufactures Y ≈ X's Y factory*", we may want to generate a rule with an additional clause: "*X manufactures Y ≈ X's Y factory*, *where Y is an artifact*". The "*where*" clause can be potentially discovered by generalizing the intersection of the *SlotY* fillers of the two relations.

## 7. ACKNOWLEDGMENTS

The authors wish to thank the reviewers for their helpful comments. This research was partly supported by Natural Sciences and Engineering Research Council of Canada grant OGP121338 and scholarship PGSB207797.

## 8. REFERENCES

[1] Anick, P.G. and Tipirneni, S. 1999. The Paraphrase Search Assistant: Terminological Feedback for Iterative Information Seeking. In *Proceedings of SIGIR-99*. pp. 153-159. Berkeley, CA.

[2] Arampatzis, A. T., Tsoris, T., Koster, C. H. A., and van der Weide, T. P. 1998. Phrase-based infase-bas retrieval. *Information Processing & Management*, 34(6):693-707.

[3] Barzilay, R., McKeown, K., and Elhadad, M. 1999. Information Fusion in the Context of Multi-Document Summarization. In *Proceedings of ACL-99*. College Park, Maryland.

[4] Dras, M. 1999. A meta-level Grammar: Redefining Synchronous TAGs for Translation and Paraphrase. In *Proceedings of ACL-99*. pp. 80-97. College Park, Maryland.

[5] Feldman, R., Fresko, M., Kinar, Y., Lindell, Y., Liphstat, O., Rajman, M., Schler, Y., and Zamir, O. 1998. Text Mining at the Term Level. In *Proceedings of the 2nd European Symposium on Principles of Data Mining and Knowledge Discovery*. pp. 65-73. Nantes, France.

[6] Hahn, U. and Schnattinger, K. 1998. Towards Text Knowledge Engineering. In *Proceedings of AAAI-98 / IAAI-98*. Menlo Park, CA. pp. 524-531.

[7] Harris, Z. 1985. Distributional Structure. In: Katz, J. J. (ed.) *The Philosophy of Linguistics*. New York: Oxford University Press. pp. 26-47.

[8] Hays, D. 1964. Dependency Theory: A Formalism and Some Observations. *Language*, 40:511-525.

[9] Hearst, M. 1992. Automatic Acquisition of Hyponyms from Large Text Corpora. In *Proceedings of ACL-92*. Nantes, France.

[10] Hindle, D. 1990. Noun Classification from Predicate-Argument Structures. In *Proceedings of ACL-90*. pp. 268-275. Pittsburgh.

[11] Iordanskaja, L, Kittredge, R., and Polguere, A. 1991. *Natural Language Generation in Artificial Intelligence and Computational Linguistics*. Kluwer. Boston, MA.

[12] Jacquemin, C., Klavans, J. L., and Tzoukermann, E. 1999. NLP for Term Variant Extraction: A Synergy of Morphology, Lexicon, and Syntax. *Natural Language Information Retrieval*, T. Strzalkowski, editor. pp. 25-74. Kluwer. Boston, MA.

[13] Larsen, B. and Aone, C. 1999. Fast and effective text mining using linear-time document clustering. In *Proceedings of KDD-99*. pp. 16-22. San Diego, CA.

[14] Lin, S. H., Shih, C. S., Chen, M. C., et al. 1998. Extracting Classification Knowledge of Internet Documents with Mining Term Associations: A Semantic Approach. In *Proceedings of SIGIR-98*. Melbourne, Australia.

[15] Lin, D. 1998. Extracting Collocations from Text Corpora. *Workshop on Computational Terminology*. pp. 57-63. Montreal, Canada.

[16] Lin, D. 1993. Principle-Based Parsing Without OverGeneration. In *Proceedings of ACL-93*. pp. 112-120. Columbus, OH.

[17] McCallum, A., Nigam, K., and Ungar, L. H. 2000. Efficient Clustering of High-Dimensional Data Sets with Application to Reference Matching. In *Proceedings of KDD-2000*. Boston, MA.

[18] Meteer, M. M. and Shaked, V. 1988. Strategies for Effective Paraphrasing. In *Proceedings of COLING-88*. pp. 431-436 Budapest.

[19] Pereira, F., Tishby, N., and Lee, L. 1993. Distributional Clustering of English Words. In *Proceedings of ACL-93*. pp. 183-190. Columbus, Ohio.

[20] Rajman, M. and Besançon, R. 1997. Text Mining: Natural Language Techniques and Text Mining Applications. In *Proceedings of the seventh IFIP 2.6 Working Conference on Database Semantics (DS-7)*.

[21] Richardson, S. D. 1997. *Determining Similarity and the Inferring Relations in a Lexical Knowledge-Base*. Ph.D. Thesis. The City University of New York.

[22] Robin, J. 1994. *Revision-based Generation of Natural Language Summaries Providing Historical Background*. Ph.D. Dissertation. Columbia University.

[23] Sampson, G. 1995. *English for the Computer - The SUSANNE Corpus and Analytic Scheme*. Clarendon Press. Oxford, England.

[24] Sparck Jones, K. and Tait, J. I. 1984. Automatic Search Term Variant Generation. *Journal of Documentation*, 40(1):50-66.